ONDERZOEKSRAPPORT NR 9113

COMMENTS ON A PAPER BY CHRISTOFIDES ET AL. FOR SOLVING THE MULTIPLE-RESOURCE CONSTRAINED, SINGLE PROJECT SCHEDULING PROBLEM

by

Erik Demeulemeester
Willy Herroelen
Wendell P. Simpson
Sami Baroum
James H. Patterson
Kum-Khiong Yang

D/1991/2376/17

COMMENTS ON A PAPER BY CHRISTOFIDES ET AL. FOR SOLVING THE MULTIPLE-RESOURCE CONSTRAINED, SINGLE PROJECT SCHEDULING PROBLEM

Erik Demeulemeester
Willy Herroelen
Department of Applied Economic Sciences
Katholieke Universiteit Leuven, Belgium

Wendell P. Simpson
Department of Quantitative Management
Air Force Institute of Technology, USA

Sami Baroum
Industrial Engineering Department
King AbdulAziz University, Saudi Arabia

James H. Patterson
Operations Management Department
Indiana University, USA

Kum-Khiong Yang
Decision Sciences Department
National University of Singapore, Republic of Singapore

COMMENTS ON A PAPER BY CHRISTOFIDES ET AL. FOR SOLVING THE MULTIPLE-RESOURCE CONSTRAINED, SINGLE PROJECT SCHEDULING PROBLEM

ABSTRACT

In a recently published article in EJOR, Christofides et al. present a depth-first search, branch-and-bound solution procedure for the multiple-resource constrained, single-project scheduling problem. While there are *many* important contributions in this paper, we show by counterexample that if the branching strategy described by the authors is used, the optimal solution might not result. Computational experience on a set of test problems appearing in the open literature is reported both with the original branching strategy suggested by the authors and a modified branching strategy that we propose. The modified strategy guarantees the determination of the optimal solution in all instances of the problem at the expense of an increase in node evaluations and average CPU time over that reported by the authors.

KEYWORDS: Project Management, Scheduling, Networks, Branch-and-Bound

1. Introduction

In a recent article in EJOR, Christofides, Alvarez-Valdes and Tamarit (CAT) (1987) present a branch-and-bound solution procedure for scheduling nonpreemptive activities of a project subject to finish-start precedence relations and multiple renewable resource constraints such that the project duration is minimized. The algorithm is based on the use of disjunctive arcs for resolving resource conflicts that develop whenever the demand for resources required by activities exceed resource availabilities during select periods of the schedule duration. Computational experience reported by the authors on a set of randomly generated project scheduling problems with up to 25 activities per project and 3 resource types per activity indicates that their procedure obtains minimum duration schedules in less computation time than does the best-first, branch-and-bound solution procedure developed by Stinson et al. (1978).

Christofides et al. make several important contributions for solving the resource-constrained, project scheduling problem. First, the authors present a quite different, yet effective framework for performing the search required to obtain and verify the optimal solution to a problem. This framework offers the advantage of narrowing the search for the optimal solution to known conflicts or conflict sets. The authors further present several unique pruning rules as well as computational experience with their usage. Finally, through the framework presented and the sophisticated use of pointers, they make very effective use of the additional memory which has been made available through recent advances in computing technology.

Unfortunately, the branching rule described by the authors excludes certain portions of the solution space from the search. This could lead to the non-detection of the optimal solution to a problem. We show by counterexample that their branching strategy does not always produce a schedule that minimizes project duration. We then propose a modification to their procedure to correct for the possibility of excluding the optimal solution. Arguments presented herein are based upon our interpretation of the algorithmic steps described in their procedure as well as a thorough study of the computer program written to implement their approach.

The modified (supplemental) branching strategy we propose is compared to the original branching strategy by solving a set of 110 project scheduling problems which are readily

available in the open literature [Patterson (1984)]. For the problems in this data set, the branching strategy proposed by the authors did not fail to find an optimal solution on any of the problems examined. However, since portions of the feasible area are not enumerated, optimality cannot be verified. The modified branching strategy, which does guarantee the selection and verification of an optimal solution, finds and verifies the same optimal solutions to these 110 test problems. These latter results are achieved through an increase in solution effort (number of nodes examined and CPU time required) on 69 of the 110 test problems.

Because of the impact of the CAT procedure for solving the constrained-resource, single-project scheduling problem (and other combinatorial problems as well), and because of typographical errors which appear in the original version of their manuscript, we also offer a revised description of their procedure to help clarify the approach. Finally, based upon recent computational experience using a framework similar to that proposed by the authors and a different branching rule [Demeulemeester and Herroelen (1990)], we conclude by indicating how an alternate branching strategy can easily be incorporated into their approach.

2. The CAT Branch-and-Bound Solution Procedure

We will use notation similar to that used in CAT:

Q	=	set of activities comprising the partial schedule
S	=	set of activities in process
H	=	set of pairs of activities involved in a direct precedence relationship
m	=	earliest time when one or more activities in S finish
p	=	current level in the solution tree
H'(p)	=	set of disjunctive arcs added at level p
C	=	set of activities which are candidates for scheduling
N	=	set of activities that cause a resource conflict
t_i	=	scheduled start time for activity i
d_i	=	duration of activity i
r_{ik}	=	per period requirement of activity i for resource k
b_k	=	amount of resource k available each period
L(i)	=	length of the longest path from the start of activity i to the end of the project
T	=	current best feasible schedule length (upper bound/incumbent)
LB0	=	a precedence based lower bound
A(i)	=	the minimal set of alternatives for delaying activity i
A	=	an alternative subset A ε A(i) of activities for delaying activity i; i.e., a subset of S whose delay would allow activity i to start within the resource limits

S*	=	a subset of S which contains the activities in S that are a member of at least one alternative A in A(i)
B(p)	=	the set of branches at level p
b(A)	=	the source node for the disjunctive arc(s) used to delay element(s) of A

The CAT procedure is a branch-and-bound process where nodes in the solution tree correspond to partial schedules Q for a subset of the activities in the project. The partial schedules represent feasible solutions for a subset of the activities satisfying both precedence and resource constraints. Partial schedules are only considered at those time instants m which correspond to the completion time of one or more project activities. At every such time instant m, a partial schedule Q contains some activities which have been completed and others which are still in process (the set S defined above). Since the objective is to minimize the length of the project schedule, the partial schedules are constructed by semi-active timetabling; i.e., each activity is started as soon as possible within the precedence and resource constraints.

At every time instant m, CAT define the set of candidate activities C as the set of activities which are not in the partial schedule and whose predecessor activities have been completed. At this juncture there is a special case to be considered. If the set of activities in process is empty and a candidate cannot be processed simultaneously with any other unscheduled activity, this non-sharing candidate can be put in process. In this special case, the algorithm then proceeds to the next time instant m, and a new candidate set is constructed.

The candidates are ordered by decreasing L(i), the length of the longest path from activity i to the end of the project, and are considered in that order. A candidate activity is put in process (added to S and Q) if it is resource feasible. If the candidate cannot be scheduled within the resources available, a resource conflict occurs. The candidate is entered into the conflict set N and the next element of C is considered. Resource conflicts produce new branching in the branch-and-bound solution tree. These new branches are possible resolutions of the resource conflict; that is, decisions about which activities are to be delayed.

One way to resolve the resource conflict is to delay the activity that causes the conflict. In order to determine other ways, CAT introduce the concept of an alternative set. For a conflict activity i the set of alternatives A(i) consists of possible minimal combinations of activities in process, the delay of which would allow the conflict activity to be scheduled.

Elements i are selected from the conflict set N in the order in which they were added (FIFO). This results in alternatives being created first for the element of N with the largest L(i). The alternatives are ordered by increasing group size. First, single activity alternatives to i are considered. Next, pairs of activities are considered. This is followed by three-activity alternatives, and so forth.

In order to delay the conflict activity i itself, CAT identify an activity a* which is to be used as the source node of a disjunctive arc leading into activity i. Activity a* is chosen as the earliest finishing activity in the set S*. In order to generate the other branches at this stage of the search process, an a' must be identified for each alternative A in A(i). Activity a' is used as the source node of the disjunctive arcs applied to delay the elements of A. Activity i is temporarily put in process by setting $t_i = m$. The a' for A is the earliest finishing activity from among the activities in (S* - A) + {i}.

The branching strategy used in the CAT procedure is to always delay candidate i (depth first search). The alternatives A(i) are considered during backtracking. Backtracking occurs whenever a schedule is completed or a branch is fathomed by comparison with a known lower bound. During backtracking, the added disjunctive arcs corresponding to the last alternative examined are removed and new arcs are added for the next alternative at the same level. Alternatives are considered in the reverse order in which they were created (LIFO). If there is no alternative left at that level, the procedure backtracks to previous level p - 1. The process is complete when level p = 0 is reached.

To determine the precedence based lower bound used by CAT (LB0), each arc added at level p in the solution tree is considered in order. One arc is added for each activity delayed. The length of the path through each added arc is then determined and LB0 is set equal to the largest of these path lengths. If LB0 fails to fathom the partial schedule, an attempt is made to strengthen the lower bound. The disjunctive arcs added at levels earlier than the current level p are searched. The algorithm attempts to locate an earlier added arc whose source node is the same as the destination node of the arc(s) just added.

If one exists, the path through these linked disjunctive arcs may be longer than LB0. If the path length through the linked disjunctive arcs is greater than or equal to T, the partial schedule is fathomed. The algorithm as described does not check for the case where three or more disjunctive arcs are linked.

The description of the CAT algorithm on page 265 of their EJOR paper can now be restated as follows:

Step 1 (Start). Let T be an upper bound on the project completion time. For every activity i, determine L(i). Set p = 0 (branching level). Put the unique starting activity 1 in process: $t_1 = 0$, Q = {1} (partial schedule); S = {1} (activities in process).

Step 2. Set m = min $\{t_i + d_i; i \in S\}$. For all $j \in S$ such that $t_j + d_j = m$, S = S - {j}. C = { } (set of candidates); N = { } (activities which cause conflicts).

Step 3 (Construction of C). For each unscheduled activity $i \notin Q$, if for all $(j, i) \in H$, $t_j + d_j \leq m$, then C = C + {i}. If C = { }, go to Step 2. Otherwise, order C by decreasing L(i). If S = { }, go to Step 11. Otherwise, go to Step 4.

Step 4 (Test of Candidates) Consider the candidates in order. For each $i \in C$, if $\Sigma_{j \in S} r_{jk} + r_{ik} > b_k$, for some k, set N = N + {i}. Otherwise, set Q = Q + {i}; S = S + {i}; $t_i = m$. If the last scheduled activity is activity n, the schedule is complete. Set T = min $\{T, t_n + d_n\}$ and go to Step 10.

Step 5. If $N = \{\}$, go to Step 2. Otherwise, go to Step 6.

Step 6 (Construction of the Set of Alternatives A(i)). Select a conflict activity $i \in N$ on a FIFO basis: $N = N - \{i\}$. Form the set of minimal delaying alternatives $A(i) = \{A \subset S \mid \Sigma_{j \in S} r_{jk} - \Sigma_{h \in A} r_{hk} + r_{ik} \leq b_k$, for all k and $(A' \in (A(i) - \{A\})) \not\subset A\}$.

The alternatives in this step are created by increasing group size (first single activity alternatives, then pairs, then three-activity alternatives, and so forth). Define S* as the set of those elements of S that are a member of at least one alternative A in A(i).

Step 7 (Identification of the Best Way of Delaying the Candidate and Each Alternative). For candidate i, find a* such that $t_{a*} + d_{a*} = \min \{t_a + d_a, a \in S^*\}$. Temporarily set $t_i = m$. For each $A \in A(i)$: find a' such that $t_{a'} + d_{a'} = \min \{t_a + d_a, a \in (S^* - A) + \{i\}\}$, and set $b(A) = a'$. Unset t_i.

Step 8 (Branching). Store m, Q and S. Delay the candidate activity i with a disjunctive arc. Set $p = p+1$ and $H'(p) = \{(a^*, i)\}$. Establish the set of branches B(p) at level p as the set of alternatives A(i) created in Step 7: $B(p) = A(i)$.

Step 9 (Lower Bound). Set $LB0 = \max\{t_j + d_j + L(i), (j, i) \in H'(p)\}$. If $LB0 < T$, strengthen the lower bound. Find at levels higher than current level p a disjunctive arc whose source node is the same as the destination node of the arc(s) just added. If one exists, the path length through the linked disjunctive arcs is determined and is assigned to LB0. IF $LB0 \geq T$, go to Step 10. Otherwise, go to Step 5.

Step 10 (Backtrack). If p = 0, STOP. Otherwise, remove all disjunctive arcs added at level p. H = H - H'(p). If B(p) = { } (all alternatives have been examined), set p = p - 1 and repeat Step 10. Select an alternative A ε B(p) with the maximum number of elements (ties are broken arbitrarily). B(p) = B(p) - {A}. H'(p) = {(b(A), j), j ε A}. Restore m, Q and S. H = H + H'(p), Q = Q - A + {i}, S = S - A + {i} and t_i = m. Go to Step 9.

Step 11 (Test for Non-Sharing Candidate Activity). Test for each i ε C. If for all nonsuccessor activities j $\notin$ Q, $r_{jk} + r_{ik} > b_k$ for some k, set Q = Q + {i}, S = S + {i}, and t_i = m: go to Step 2. Otherwise, go to Step 4.

3. Counterexample to the Optimality of the CAT Procedure

Consider the activity-on-the-node network given in Figure 1(a). The numbers above each node denote the fixed activity durations. The numbers below each node denote the per period resource requirement for the single resource type involved in completing this project. The resource has a constant availability of 5 units per time period. The unique optimal schedule for this project is represented in Figure 1(b) and has a duration of 7 periods. The CAT procedure yields the nonoptimal schedule shown in Figure 1(c). The nonoptimality of the CAT algorithm is caused by the branching strategy employed, as we now describe.

==========

Insert Figure 1

==========

The CAT procedure schedules the dummy start activity 1, yielding the partial schedule Q = {1} and the set of activities in process S = {1}. Step 2 of the procedure yields m = 0 and S = { }. The set of candidates is initialized as C = { }, and the set of conflict activities is N = { }. Step 3 of their procedure then generates the set of candidates ordered in decreasing order of the remaining critical path length: C = {2, 5, 4, 3}. Continuing with Step 4, S = {2, 5, 4}. The set of conflict activities is updated as N = {3}. In Step 6 of their procedure, conflict activity i = 3 is selected yielding N = { }. The minimal set of alternatives is generated: A(3) = {{5}, {4}}. The set S* is defined as S* = {5, 4}.

Step 7 then identifies a* = 5 for candidate i as the earliest finishing activity in the set S*. This yields the extra precedence relationship (5, 3) corresponding to the first node constructed at level p = 1 in the branch and bound solution tree. The a' for each alternative are then generated as follows. Activity i = 3 is temporarily put in process and the a' for alternative A = {5} is found as a' = 3, the earliest finishing activity from among

the activities in (S* - A) + {i}. For the second alternative A = {4}, a' = 3. This yields two additional nodes at level p = 1 of the solution tree corresponding to the added precedence relationships (disjunctive arcs) (3, 5) and (3, 4).

As can be seen from the unique optimal schedule given in Figure 1(b), activity 5 does not precede activity 3 in the optimal solution. Nor does activity 3 precede activities 4 or 5. Continuing the search from any of the three nodes at level p = 1 can never lead to the optimal solution. Restricting the search to elements of S* clearly leads to the determination of a nonoptimal solution for this example problem.

4. A Modified Branching Strategy

The difficulty inherent in the CAT branching strategy occurs in Step 7 of their procedure. The counterexample (Figure 1) demonstrates that the entire set S, rather than S^*, must be considered when creating disjunctive arcs. Restricting the search for source nodes of the disjunctive arcs to elements of S^* effectively fathoms partial schedules that should not be fathomed. The counterexample shows that the elements of (S - S^*) cannot be ignored when creating disjunctive arcs.

One remedy for altering the branching strategy is to substitute S for each occurrence of S^* in Step 7 of their procedure. This modification ensures that no partial schedule is prematurely fathomed in their approach. For the counterexample, use of the modified branching strategy changes the p = 1 level partial solution nodes to:

1) Delay candidate activity 3 with disjunctive arc (2, 3).
2) Delay candidate activity 5 with disjunctive arc (2, 5).
3) Delay candidate activity 4 with disjunctive arc (2, 4).

The modified branching strategy finds the optimal schedule depicted in Figure 1(b) using the set of disjunctive arcs {(2, 3), (6, 4), (6, 3), (5, 4) and (3, 8)}.

The modified strategy does not alter the total number of alternatives (nodes) created at each level in the solution tree, but it can increase the total number of node evaluations

required to find the optimal solution. When creating the disjunctive arcs for each node, the revised strategy will select an activity $j \in (S - S^*)$ as a source node of a disjunctive arc whenever $t_j + d_j < \min\{t_a + d_a , a \in S^*\}$[1]. When this happens, the path length assigned to LB0 will be shorter than if the source nodes had come only from S^*. It may happen that nodes that would have been fathomed prematurely by LB0 under the original branching strategy will not be fathomed under the modified strategy.

Results obtained using the modified branching strategy are compared to results obtained using the original branching strategy in Table 1. Worst case results for example problems in this data set are given in Table 2. The original branching strategy did not fail to find the optimal solution for any of the 110 test problems. The modified branching strategy leads to an increase in solution effort on 69 of the 110 test problems. In terms of the number of node evaluations[2], the modified strategy results in a mean increase of 9%, with a worst case increase of 125% on Problem 31. In terms of CPU seconds, the mean increase is 8%, with a worst case increase of 191% on Problem 12. These problems are further described by Patterson (1984).

==========

Insert Tables 1 & 2

==========

[1] Or $(S^* - A) + \{i\}$ when creating disjunctive arcs for the alternatives.

[2] A node evaluation occurs at the end of Step 9 of their procedure whenever LB0 < T.

5. An Alternate Branching Strategy

Demeulemeester & Herroelen (1990) describe an effective optimizing procedure for resolving resource conflicts in project schedules. At level p of their solution tree, they define a delaying set D(p) as the set of all minimal delaying subsets D_q of activities, either in process or eligible to start, the delay of which would resolve the resource conflict. The search is restricted using this approach to minimal delaying alternatives; i.e., delaying alternatives which do not contain other delaying alternatives as a subset. The delay of a subset of activities $D_q \in D(p)$ is then introduced by defining a set of additional precedence relations $G_q = \{(j, i)\}$ using as the predecessor activity for every activity $i \in D_q \in D(p)$ the earliest finishing activity j that is either in progress or eligible to start at time m, that is not currently delayed (ties are broken arbitrarily).

In the example problem of Figure 1, this branching strategy leads to the following result. Starting with the dummy activity 1 yields the unordered set of candidates C = {2, 3, 4, 5}. Temporarily starting all candidate activities leads to a resource conflict. It is sufficient to release 2 units of the single resource in order to resolve the resource conflict. The delaying set is generated as D(1) = {{3}, {4}, {5}}. For each delaying alternative, activity 2 is found to be the earliest finishing activity yielding the added precedence relations (disjunctive arcs) (2, 3), (2, 4) and (2, 5). These added precedence relations define three branching alternatives at level p = 1 in the solution tree. This depth first branching strategy can be incorporated into the CAT procedure or can be used as described by Demeulemeester and Herroelen in their approach. Excellent results have been obtained on these same 110 test problems using this approach.

6. Conclusion

In this note we have revised the presentation of the CAT procedure for solving the multiple resource-constrained, single-project scheduling problem. A 9-activity counter-example problem is used to demonstrate that the branching strategy originally implemented in their approach may yield nonoptimal schedules (solutions). We demonstrated that the construction of additional precedence relations (disjunctive arcs) used in the CAT procedure for delaying activities is too restrictive. We then suggest a modified branching strategy and evaluate it on a set of 110 test problems taken from the literature. This modified strategy results in an increase in solution effort, but with the guarantee of an optimal solution. Finally, we describe an alternative depth first branching strategy which has been successfully implemented in a branch-and-bound solution procedure with impressive results on this same set of test problems.

In closing, we wish to note that it has not been our intention to suggest that major modifications are needed in the CAT procedure. Rather, our objective has been to contribute a modification to what appears, by the authors' reported computational results, to be a very effective procedure for solving the constrained-resource, project scheduling problem. This modification does, however, guarantee the optimality of the solutions obtained with this procedure.

References

Christofides, N., Alvarez-Valdes, R. and Tamarit, J.M. (1987), "Project Scheduling With Resource Constraints : A Branch-and-Bound Approach", European Journal of Operational Research, 29, 262-273.

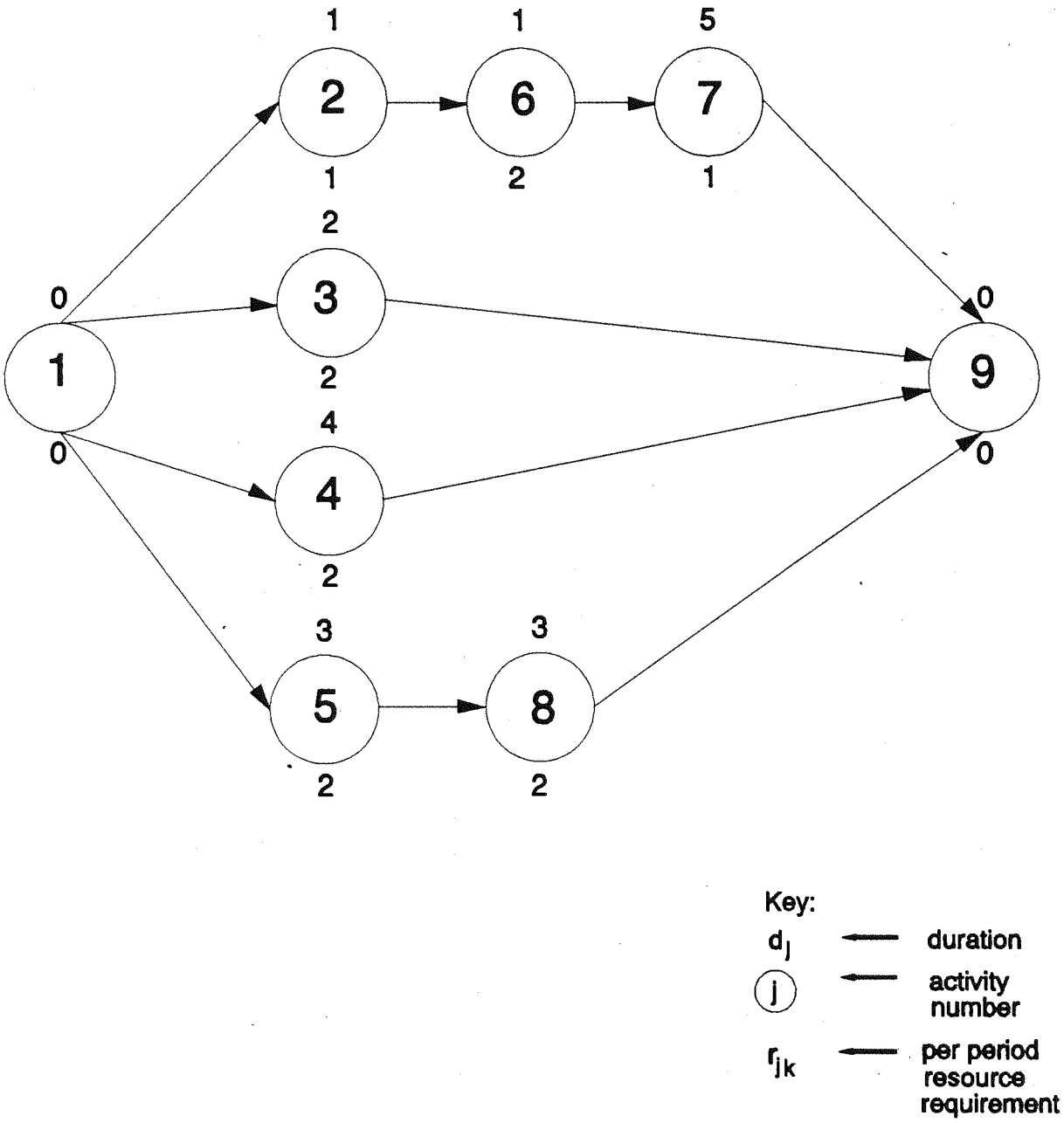
Demeulemeester, E. and Herroelen, W. (1990), "A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problem", Paper presented at the Second International Conference on Project Management and Scheduling, Compiègne, June 20-22, 1990.

Patterson, J. H. (1984) "A Comparison of Exact Approaches for Solving the Multiple Constrained Resource, Project Scheduling Problem", Management Science, 30, 854 - 867.

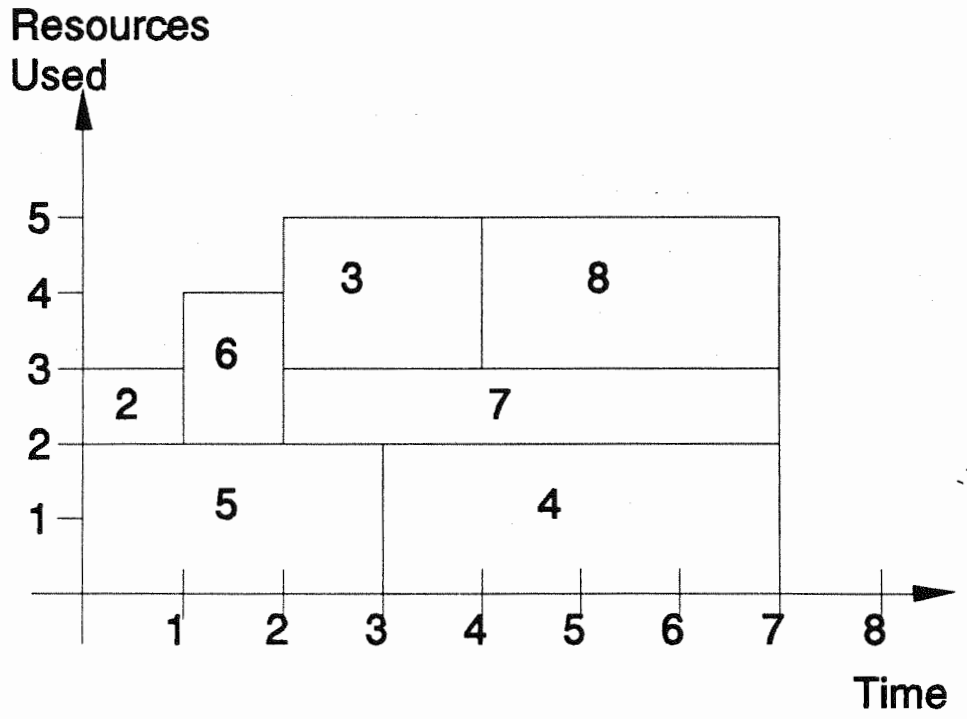
Stinson, J.P., Davis, E.W. and Khumawala, B.M. (1978), "Multiple Resource-Constrained Scheduling Using Branch-and-Bound", AIIE Transactions, 10, 252-259.

Figure 1. Counterexample Showing a Nonoptimal Solution Obtained by the CAT Algorithm

(1a) Activity Network



(1b) Unique Optimal Schedule



(1c) Schedule Obtained by CAT

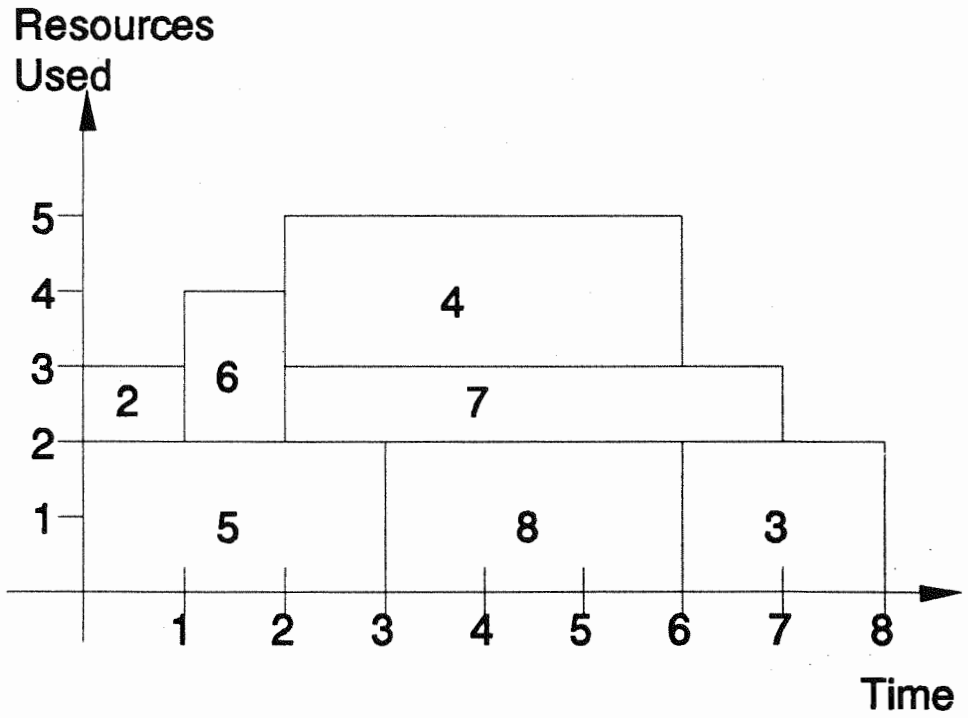


Table 1 Summary Performance of the Modified Branching Strategy vs the Original Branching Strategy

	NODE EVALUATIONS			CPU SECONDS*		
	Original Strategy	**Modified Strategy**	**Percent Increase**	**Original Strategy**	**Modified Strategy**	**Percent Increase**
	All Problems (n = 110)					
Median	823	972	18%	4.5	5.1	14%
Mean	8,630	10,012	16%	48.4	56.8	17%
Std. Dev	28,497	33,790	19%	163.2	186.5	14%
	Problems with Increased Node Count (n = 69)					
Median	1,753	2,230	27%	9.0	11.3	26%
Mean	13,250	15,452	17%	75.1	88.5	18%
Std Dev	35,147	41,698	19%	201.3	229.6	14%

*IBM 3090 CPU time, in seconds.

Table 2. Select Test Problems With Worst Case Performance for Modified Strategy

		NODE EVALUATIONS			CPU SECONDS*		
Prob. Title	**No. of Activities**	**Original Strategy**	**Modified Strategy**	**Percent Increase**	**Original Strategy**	**Modified Strategy**	**Percent Increase**
12 DUPONT	23	188	379	102%	0.300	0.890	197%
13 SEE-2	22	36,260	102,890	184%	285.023	807.330	183%
31 C2C	22	470	1,057	125%	6.590	15.830	140%
101 C4CM**	51	22,992	23,700	3%	1,433.120	1,497.500	4%

*IBM 3090 CPU time, in seconds.
**Problem included because of time taken for solution.